REMARKS

Claims 20-21 have been added. Claims 1-21 are pending. Reconsideration and allowance are respectfully requested in view of the following remarks.

Claim 11 has been amended in line with the Examiner's suggestion to replace "instructions" with "instruction". This amendment is made to clarify grammar and is not viewed as narrowing. No other amendments have been made to claims 1-19.

The present invention is concerned with a new type of "lister". A lister takes an object code sequence as an input and through a conversion process known as disassembling produces a source code listing. Figure 2 shows that a source code module 1 is converted by an assembler/compiler 2 into an object code module 3. Various object code modules 3 are then linked by a linker 4 to form the executable program code 5. The present invention resides in the lister 8 which draws from the object code module 3 and other inputs such as from library object files 6 or the executable program code 5 to produce the original source code. Generation of the original source code is especially useful for testing and/or debugging code (see page 7, third paragraph).

Figure 5 shows an object module 3 having section data (01, 02, 03) and an associated relocation section (R1, R2, R3). Figure 6 shows the general components of a lister which include a data reader 11 for reading the section data and a relocation reader 16 for reading the associated relocations 20. The lister further comprises a directive processor 30, an expression calculator 32 and an event calculator 36. Depending on the type of the relocation instruction as determined by the relocation reader 16, the relocation instruction is passed to one of the units 30,

32 or 36 for processing (see page 17 lines 17 –25). Thus, the lister acts on each location of section data in turn and uses a relocation reader to determine if there are any relocations associated with that location of section data read by the data reader 11. If there is no associated relocation, then the disassembler carries out a conventional disassembler operation. However, if there is an associated relocation then the relocation is passed to one of the units 30, 32 or 36 depending on the type of relocation which is determined by the relocation reader 16.

Claims 1-4, 6-17 and 19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Brooks.

Cahill is not concerned with a lister (or lister methods). Rather, Cahill is concerned with transforming an object code module into a form which can be readily manipulated (i.e., into a platform-independent assembly code as represented by the PIAI module 230). Cahill does not disclose anywhere generating source code from an object code module, or more importantly the mechanism or methods for doing this. That is, there is no disclosure in Cahill of a mechanism or method for reading relocation instructions. The Examiner admits as much and since Cahill is not even concerned with generating source code it is wholly irrelevant to the claimed invention. For this reason alone, the rejection is without merit and should be withdrawn.

Cahill discloses that the PIAI module (i.e., the transformed code) comprises a linked list of individual assembly code instructions (see column 7, lines 40 to 42), but assembly code is not source code. Further, Cahill describes that each instruction in the linked list comprises pointers to tags used in the instruction (see column 7, lines 49 to 50) and that these tags are provided from

the platform independent object format (PIOF) module. However, these are machine code instructions and are not material to the claimed invention.

The Examiner admits that Cahill does not describe reading relocation instructions identified for each location in the section data, but argues that Cahill discloses a disassembling process to recompose the original object code and refers to Figures 4 and 5 and column 9, line 45 to column 10, line 50 of Cahill. Cahill is concerned with a number of transformation steps including decomposing an original object module 130 into a platform independent module 220 and transforming this into a more easily manipulated format, i.e., by generating a PIAI module in assembly code. The assembly code can then be modified using a modifier 260 to generate a new PIAI module 240- which is turn is reassembled 270 to form a new PIOF module 220- and this in turn is finally recomposed 280 to form a new object module 150-.

The Examiner however admits that Cahill does not describe "generating a source code listing, such listing being derived from said relocation instructions and additional relocation information", but instead asserts that Brooks discloses unlinking the object program code into source code using additional information such as memory address reallocated values, which the Examiner asserts is equivalent to relocation instructions of the present invention. However, Brooks makes no mention of "memory address reallocated values", but instead discloses variable tag names and a symbol table 70. An understanding of these concepts in the context of Brooks will help understand how the present invention is distinguished over the art..

Brooks is concerned with a compiler which draws code fragments from an instruction table, wherein instructions in the source program are read and matched up with a particular code

fragment in the instruction table, and code fragments are assembled to form an object program. This compilation from source code to an object program is a two stage process in Brooks. First, an unlinked object program module 56 is created from the code fragments identified in the instruction table. This unlinked object 56 comprises tag values (i.e., identifiers) for the operands "a" "b" [of the instruction "add A, B, A"], because physical addresses in memory have not yet been established for these variables, which will be done in a subsequent linking step by the linker 66 which forms the final object program 24. That is, the linker 66 receives the unlinked object program module 56 which identifies each of the variable tag names (A and B) and finds the physical address locations in memory and inserts these physical addresses into the memory image object program 24 (see column 7, lines 45 to 51).

During the linking step, the linker 66 creates a symbol table 70 which relates each symbol name (A and B) to its physical address. The symbol table is said to be attached to the object program for use in decompilation (see column 7, line 61 to column 8, line 2). The step of decompilation includes reading the physical addresses, looking up those addresses in the symbol table and providing the associated tag names (A, B). That is, the tags and symbol table of Brooks are merely used for establishing a link between a physical address location of an operand in memory and the associated variable name used in the original source code. However, a symbol table establishing a link between variable names and their physical addresses, is not the same as relocation instructions of the present application. There is no disclosure of relocation instructions in Brooks and more importantly of the claim structure of the present invention, i.e., "having an object code sequence comprising section data having associated therewith a

relocation section including at least one relocation instruction". Thus, an operand variable name and location is quite clearly not the same as a relocation instruction, and more importantly there is no disclosure in Brooks of a mechanism for reading said relocation instruction for determining each type of relocation instruction. Applicant accordingly asserts that claims 1 and 11 of the present application are distinguished over Cahill and Brooks.

Applicant further asserts, in view of the foregoing discussion, that the Examiner's combination of Cahill and Brooks is incorrect since the references lack any motivation to a skilled person to make such a combination. Although Brooks does describe a decompilation process for recovering source code from object code, Cahill is not concerned with this at all and therefore a skilled person would not have been prompted to look at Cahill. Applicant respectfully submits that the Examiner has made the combination improperly by exercising hindsight.

Claim 5 was rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Brooks and Brandes. Also, Claim 5 was rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Noda and Brandes. Applicant traverses and asserts that claim 5 is patentable for at least the reasons discussed above with respect to claim 1.

Claim 18 was rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Brooks and Clarke. Also, Claim 18 was rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Noda and Clarke. Applicant traverses and asserts that claim 5 is patentable for at least the reasons discussed above with respect to claim 11.

Claims 1-4, 6-17 and 19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Brandes in view of Noda. Applicant respectfully traverses. This rejection is not well stated by the Examiner. The discussion in Section 5 (see, page 6, et seq.) of the Office Action focuses on Cahill and not Brandes. The Examiner has failed to discuss at all how Brandes meets the claimed limitations. Accordingly, applicant submits that the Examiner has failed to make out the prima facie case and this rejection should be withdrawn.

To the extent the rejection is instead Cahill in view of Noda, applicant again traverses. Arguments with respect to Cahill have been given above and it is applicant's position that Cahill is not relevant to the present application. The Examiner asserts that Cahill does not disclose reading said relocation instruction identified in the section data, but says that "this limitation has been addressed in claim 1 above". Applicant does not understand this reasoning, since this limitation is in claim 1, but this limitation is not disclosed in Cahill and the Examiner does not assert that Noda discloses this feature either. Therefore, it is not apparent what reference the Examiner asserts discloses this feature of claim 1.

The Examiner asserts that although Cahill does not disclose the feature of "generating the source code listed for that location in the section data, the source code listing comprising source code from which object code at that location was derived with said additional information derived from the relocation instruction", that Noda discloses this feature. Again, applicant disagrees with this argument since Noda appears to be similar to Brooks in dealing with providing a symbol table of sorts which gives labels through reference addresses and substitutes

reference addresses with the corresponding labels. This substitution of address information with labels and/or vice versa is not the same as relocation instructions.

To underline this distinction between relocation instructions and a symbol table, one only needs to look at Cahill which clearly shows that the object module 130 comprises separate sections of a symbol table 304, as opposed to relocation instruction sections 305 and 306. These sections are all amalgamated into the character table 318 of the generated PIOF module 220 of Cahill. That is, relocation instructions are clearly different from a symbol table. In the context of both Brooks and Noda, which are concerned with assigning operand variables to physical addresses this is merely a standard symbol table as indicated by 304 in Cahill, which is entirely different from relocation instructions.

Withdrawal of the rejection based on Brandes/Cahill in view of Noda is accordingly requested.

New claims 20-21 have been added. These claims are directed to determining relocation type. Since Cahill does not disclose any mechanism for reading relocation instructions, Cahill certainly does not disclose a reading mechanism able to determine a type of the relocation instruction. From this type information, one may derive additional information. Additionally, this information may be derived by different relocation processing mechanisms, such as the directive processor 30, expression calculator 32 or the event calculator 36 of the present application, depending on type.

In view of the above, it is believed that this application is in condition for allowance, and such a Notice is respectfully requested.

Respectfully submitted,

JENKENS & GILCHRIST, P.C.

Date: 3/23/04

Andre M. Szuwalski
Registration No. 35,701

1445 Ross Avenue, Suite 3200
Dallas, Texas 75202-2799
(Direct) 214/855-4795
(Fax) 214/855-4300